MORRISON | FOERSTER

425 MARKET STREET
SAN FRANCISCO
CALIFORNIA 94105-2482

TELEPHONE: 415.268.7000
FACSIMILE: 415.268.7522

WWW.MOFO.COM

MORRISON & FOERSTER LLP

NEW YORK, SAN FRANCISCO,
LOS ANGELES, PALO ALTO,
SACRAMENTO, SAN DIEGO,
DENVER, NORTHERN VIRGINIA,
WASHINGTON, D.C.

TOKYO, LONDON, BRUSSELS,
BEIJING, SHANGHAI, HONG KONG

November 9, 2011

Writer's Direct Contact
415.268.7455
MJacobs@mofo.com

The Honorable William H. Alsup
United States District Court, Northern District of California
450 Golden Gate Ave., Courtroom 8, 19th Floor
San Francisco, CA 94102

Re: *Oracle America, Inc. v. Google, Inc.*, Case No. 3:10-cv-03561-WHA

Dear Judge Alsup:

The Court asked Google to analyze concrete examples from at least two APIs to support Google's proposal for an advance ruling on the 37 APIs through analytic dissection. Apparently realizing it isn't seeking analytic dissection after all, Google nowhere performs this analysis.

Instead, with respect to every single aspect of the APIs it discusses, all Google says is that it had to implement those precise elements in order to achieve *compatibility*. Google already made this argument on summary judgment and lost. Google is not asking this Court to engage in an analytic dissection analysis of the type conducted in *Apple v. Microsoft*. It is asking to reargue its failed motion.

While Google frames its précis as a request for the Court to make an advance determination of copyrightability, it actually is asking for something different. Google's professed desire for compatibility (which only lasts until Google says it doesn't actually want to be fully compatible) goes more to its affirmative defense of fair use—a defense the Court has already held raises factual issues for trial—than to the copyrightability of the APIs themselves.

Google does not explain why its request will save the Court time. It will not. The Court has already held that the API design specifications are not unprotectable methods of operation. Oracle expects to prevail on copyrightability, but even if Google were to show that some or all elements of the API specifications are subject to merger or *scenes a faire*, Oracle would still present witnesses to show Google is liable for infringement because it made "virtually identical" copies.

I. THE JAVA API DESIGN SPECIFICATIONS

Copyright law protects expression in software design, including the selection and structure of software elements. The 37 Java API design specifications at issue represent years of creative effort in designing software that offer powerful features to developers. An API designer faces many creative choices: for example, which classes and interfaces to include (and whether a particular element should be a class or an interface), which fields and methods to include in each class or interface, the relationship of each element to other elements, and how to name and organize all of the elements in an elegant and easy-to-learn way. The choice of what to leave out—which classes should be left for programmers to implement, which fields and methods should be declared "private" and therefore part of a particular implementation but not the API—is creative as well. If the designer includes too little in an API, the developers will not have the tools and flexibility they prefer. If the designer includes too much, the APIs become overwhelming and nonintuitive.

Two examples—java.util and java.nio—reflect significantly different design choices and illustrate the originality and creativity of the Java authors. The java.util package specification defines 49 classes, 16 interfaces, 1 enumeration, and 20 exceptions. Its diverse classes provide, among other things, calendar functions, random number functions, and collection manipulation functions—none of which have any necessary relationship to one another. (There is no technical or functional requirement that the calendar and collection classes be in the same package, for example.) Yet the API designers decided that these unrelated classes should be put in the same package. *See* http://download.oracle.com/javase/1.5.0/docs/api/java/util/package-summary.html.

The API designers took a different approach to java.nio. The term "nio" stands for "New Input/Output," and was born of a project led by Sun to create an improvement over the then-existing Java I/O API, the java.io package. Although Sun decided that its new API should define classes implementing buffers, character sets, channels, and selectors, Sun chose to put only the

buffer-related classes into java.nio, and put the other classes into subpackages. Sun could have done it much differently: it could have augmented the java.io package to include all the new classes, or it could have put all the new classes into java.nio, rather than making separate subpackages. These choices were creative, not driven solely by function. In choosing to copy the selection and arrangement of the java.nio and java.util packages (as well as 35 other core Java API design specifications), nearly all in their entirety, Google infringed on a massive basis.

The API elements are arranged in a primarily hierarchical relationship, based on their inheritance from the interfaces they implement and the classes they extend. An example of class inheritance—and of Google's copying of inheritance— is shown in these excerpts from the java.util specifications for Java and Android for the class "Hashtable":

Java SE 5.0 java.util API specification	Android 2.2 (Froyo) java.util API specification
java.util **Class Hashtable<K,V>** java.lang.Object └ java.util.Dictionary<K,V> └ **java.util.Hashtable<K,V>** **All Implemented Interfaces:** Serializable, Cloneable, Map<K,V>	public class **Hashtable** extends Dictionary<K, V> implements Serializable Cloneable Map<K, V> java.lang.Object ↳ java.util.Dictionary<K, V> ↳ java.util.Hashtable<K, V>

According to Oracle's java.util API specification, the Hashtable class is a subclass of Dictionary, and implements three interfaces: Serializable, Cloneable, and Map, which are found in the java.lang, java.io, and java.util packages, respectively. This organization is hardly intuitive or preordained, but Google copied it. Android's Hashtable class is likewise a subclass of Dictionary, and implements the same three interfaces, with the same three names, found in the same three corresponding Android packages.

The exceptions defined in the java.nio specification provide another example of the hierarchical choices made by the designers. Whereas BufferOverflowException was chosen to

be a subclass of java.lang.RuntimeException, InvalidMarkException was chosen to be a subclass of java.lang.IllegalStateException. In making these choices, the API designers expressed to developers that these two exceptions were different kinds of errors that should be categorized differently. The choice represents a creative expression with substantive meaning to developers, not something inalterably driven by function. And Google copied the organization and the names—not just in a substantially similar manner, but in a virtually identical form:

Java SE 5.0 java.nio API specification	Android 2.2 (Froyo) java.nio API specification
java.nio **Class BufferOverflowException** java.lang.Object └ java.lang.Throwable └ java.lang.Exception └ java.lang.RuntimeException └ **java.nio.BufferOverflowException**	public class **BufferOverflowException** extends RuntimeException java.lang.Object ↳ java.lang.Throwable ↳ java.lang.Exception ↳ java.lang.RuntimeException ↳ java.nio.BufferOverflowException
java.nio **Class InvalidMarkException** java.lang.Object └ java.lang.Throwable └ java.lang.Exception └ java.lang.RuntimeException └ java.lang.IllegalStateException └ **java.nio.InvalidMarkException**	public class **InvalidMarkException** extends IllegalStateException java.lang.Object ↳ java.lang.Throwable ↳ java.lang.Exception ↳ java.lang.RuntimeException ↳ java.lang.IllegalStateException ↳ java.nio.InvalidMarkException

A third example is the Java Collections Framework, which is defined in the java.util API specification. The organization and hierarchy of the collections classes in Java is very different from that in other object-oriented languages, such as Smalltalk and C++, although they serve the same function. Google copied this unique organization as well. (*See* ECF No. 341, Ex. 1 ¶¶ 193-199 & Ex. 4.)

These are only three examples of expressive choices in the API specifications. Google repeatedly acknowledges it intentionally copied from Oracle's copyrighted API specifications,

including the selection, arrangement, and structure of the API elements. (*See* Google Précis at 7-15.) This copying includes thousands of names, classes, subclasses, interfaces, fields, methods, and exceptions. In the case of the java.util package alone, Google copied 49 classes, 16 interfaces, 20 exceptions, 85 fields, and 369 methods, and the structural relationships among them, as well as their structural relationships to the elements of other packages.

The selection, arrangement, and structure of API elements are an expression of the ideas underlying the API specifications. The creative effort that produced the specifications readily surpasses the "minimal" amount of originality required for copyrightability. *See CDN Inc. v. Kapes,* 197 F.3d 1256, 1260-61 (9th Cir. 1999). Google cannot seriously contest this. Both Google's engineers and Google's copyright expert have acknowledged that designing an API requires skill and creativity. As Google's expert testified in deposition:

> Q. Okay. And what are the elements of creativity involved in designing APIs?
>
> A. I think that those are the same things that are the elements that – of creativity that go into designing. Other things related to software. And that's kind of understanding how the software is meant to function.
>
> And if you have a good understanding of how the software is going to function, and if you know the intended audience, that helps you to create things that that audience will like, will be able to use.
>
> So the creativity is kind of understanding how the thing you're going to design is going to be used. And understanding the audience or the libraries that you're going to develop an API for, the creative aspects come into can you actually get at the needs or the wants or the usage of who your audience is.

(Astrachan Dep. 128:23-129:14.) Dr. Astrachan also testified:

> Just as it's hard to find people that are really good at anything that's hard, whether it be, you know, being an artist, a football player, a concert violinist. Those things are hard. This [API design] is something that's hard in the same way. (*Id.* at 128:9-13.)

What distinguishes a great API from a good API is not its function, but its elegance and ease of use, its intuitive appeal, and its communication of substantive meaning and direction to

developers. The authors of the Java APIs aimed for all three, and the success of the Java platform is a testament to their creativity.

II. GOOGLE IS ASKING THE COURT TO DECIDE ISSUES THAT IT SHOULD BE REQUIRED TO PROVE AT TRIAL

Google's précis does not propose that the Court engage in the type of analytic dissection of individual elements the district court performed in *Apple v. Microsoft*. Instead, despite being directed to provide specific examples, Google contends it should be given a blanket excuse for its copying because it wanted to make Android's APIs compatible with the 37 Java APIs it copied. (*See, e.g.,* Google Précis at 7-10, 12-15.) Google already made this sweeping argument in its summary judgment motion, and the Court rejected it. (*See* ECF No. 260, Google SJ motion at 14-17, 19-22; ECF No. 433, Summary Judgment Order ("SJ Order") at 9, 12-13.) Now Google tries again, suggesting the parties should "update the briefing and declarations they submitted in connection with Google's copyright summary judgment motion…." (Google Précis at 5.) Google is not entitled to a second bite at the summary judgment apple.

Moreover, while Google frames its précis as asking the Court to decide the copyrightability of the selection, arrangement and structure of the 37 API specifications (Google Précis at 1), in reality it is not. Google's Ninth Circuit authorities, *Sega* and *Sony*, both addressed the defendants' desire for compatibility in the context of evaluating *fair use*, not copyrightability. *See Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1520-28 (9th Cir. 1993); *Sony Computer Entm't, Inc. v. Connectix Corp.*, 203 F.3d 596, 602-09 (9th Cir. 2000).

Google should be required to try to prove its affirmative defense of fair use at trial if it can, and its desire for compatibility is, at best, only a part of one of at least four statutory factors that should be considered. (*See* SJ Order at 12 (quoting 17 U.S.C. § 107).) The Court has already found that Google's fair use defense raises factual issues that preclude summary judgment in Google's favor. (*Id.* at 12-13.)

But even if Google had proposed that the Court perform a detailed analysis of individual elements of the API specifications in advance of trial, that would not be a useful exercise here. Even Google acknowledges analytic dissection is "helpful," rather than mandatory. (Google Précis at 3 (quoting *Apple Computer, Inc. v. Microsoft Corp.,* 35 F.3d 1435, 1443 (9th Cir. 1994).) Analytic dissection is not required here, and certainly not in advance of trial, *where Google admits it engaged in wholesale, deliberate copying* of literal and non-literal elements of the 37 API specifications, and where there are no issues of pre-existing license that pervaded *Apple v. Microsoft*. As the Tenth Circuit has noted, "Not every case requires an extensive abstraction-filtration-comparison analysis. Rather, 'the appropriate test to be applied and the order in which its various components are to be applied . . . may vary depending upon the claims involved, the procedural posture of the suit, and the nature of the [works] at issue.'" *Mitel, Inc. v. Iqtel, Inc.*, 124 F.3d 1366, 1372 (10th Cir. 1997) (citation omitted) (omissions in original); *see also* 4-13 *Nimmer on Copyright §* 13.03[A][1] (explaining need for extensive analytic dissection and similar tests arises only when there is "comprehensive nonliteral similarity").

Oracle's case is based on Google's massive wholesale copying of literal and nonliteral elements, not copying of only nonliteral elements or a mere handful of literal elements. Substantial similarity will not turn on whether some of the elements individually could be filtered out. Google acknowledges that for nearly all of the 37 APIs *it copied everything*, so substantial similarity—and even virtual identity—is a given. *See id.* at § 13.03[A][2] ("Where there is literal similarity (virtually, though not necessarily, completely word for word) between plaintiff's and defendant's works … it is not necessary to determine the level of abstraction at which similarity ceases to consist of an 'expression of ideas,' because literal similarity by definition is always a similarity as to the expression of ideas.").

Google is also incorrect in suggesting that courts decide copyrightability as a matter of law regardless of whether there are underlying factual issues for trial. *See, e.g.*, *Yue v. Chordiant*

Software, Inc., No. C 08-00019 JW, 2009 U.S. Dist. LEXIS 118824, at *7 (N. D. Cal. Dec. 21, 2009) (and cases cited therein) (finding triable issue as to whether plaintiff's work was sufficiently original to warrant copyright protection). As Oracle stated in its trial brief, it believes that, after hearing the evidence presented at trial, the Court will decide copyrightability in its favor as a matter of law, but the Court may determine there are threshold questions of fact for the jury. (*See* ECF No. 536, Oracle Trial Brief at 23-24.)

Finally, Google's argument that it is "crucial" for it to have further clarification on what Oracle is claiming was copied (Google Précis at 3) is meritless. Google copied the elements of the 37 API design specifications and the relationships among them, as illustrated in section I above, and reproduced them in its own specifications. It knowingly created a derivative work by basing its class library source code on the API design specifications, adopting Oracle's selection, arrangement, and structure. Google acknowledges this copying. (*See, e.g., id.* at 7-15.)

III. GOOGLE'S COMPATIBILITY ARGUMENT IS LEGALLY INCORRECT, AND PROVIDES NO BASIS FOR A DETERMINATION BEFORE TRIAL

Google's proposal for an advance determination of its compatibility argument rests on a false factual premise and a misreading of the applicable decisions.

First, Android is not actually compatible with Java. Google fragmented Java by implementing less than the full number of the Java API packages ("subsetting"), adding its own APIs ("supersetting") and, in some cases, only supporting portions of the classes within a Java API package. (*See* ECF No. 341-2, Mitchell Copyright Opp. Report ¶¶ 61-68, 101-113.) For example, Google chose to implement only 2 of the 16 classes in the java.awt.font package. As a result of Google's copying of only the parts of Java it wanted, Java programs that rely on the elements Google did not implement will not run on Android, and Android programs will not run on Java-compliant virtual machines. If Google had truly intended existing Java programs to be

compatible with Android, it would have implemented the complete set of Java APIs and certified Android as Java-compatible using Oracle's Technology Compatibility Kit.

Google argues that "to ensure compatibility *with the Oracle API packages that Google did implement in Android*, Google *had* to use the same selection of API elements, and the same hierarchical arrangement and structure for those elements." (Google Précis at 8 (emphasis in original).) The Court should reject this circular, selective compatibility argument—which even Google recognizes sounds "tautological." (*Id.* at 14.) In essence, Google is saying that it wanted to copy certain features, so its infringement is justified because in order to achieve compatibility with these features, Google needed to copy them.

Google's claim that it copied only what it had to in order to achieve compatibility is also false. (*Id.* at 7-8.) In many instances Google copied features that were not required for compatibility. For example, while Google may claim it needed to copy the types and orders of parameters in a method signature in order to achieve compatibility, it was not necessary for Google to use the same parameter names. (*See* ECF No. 391, Astrachan Rebuttal Report ¶ 42.) Where Google did copy Oracle's parameter names, it did so because it wanted to. For example, the table below, excerpted from Prof. Mitchell's Copyright Opening Report (ECF No. 341, Ex. 1 ¶ 205), shows method signatures and parameters for the class java.nio.IntBuffer. Google copied two of Oracle's parameter names, "index" and "dst," but modified two others, adding a prefix to "offset" and replacing "length" with "intCount." Google could have used different names in place of "index" and "dst" as well, but instead Google copied Oracle's name selection.

Methods in java.nio.IntBuffer (Java version)	Methods in java.nio.IntBuffer (Android version)
abstract int get(int index)	abstract int get(int index)
IntBuffer get(int[] dst, int offset, int length)	IntBuffer get(int[] dst, int dstOffset, int intCount)

Google's argument is also legally incorrect. Google misreads *Sega* and *Sony*. Both were "intermediate copying" cases that examined whether it was fair use to reverse engineer a

copyrighted program where "disassembly is the only way to gain access to the ideas and functional elements embodied in a copyrighted computer program." *Sega Enters.*, 977 F.2d at 1527-28. The Ninth Circuit held that this intermediate copying "must have been 'necessary' to have been fair use." *Sony Computer*, 203 F.3d at 603 (citing *Sega*, 977 F.2d at 1524-26). *Sega* and *Sony* did not hold, as Google suggests, that there is no infringement "if the implementing code was not copied." (Google Précis at 6.) Rather, because the plaintiffs in those cases did not accuse the defendants' finished products of infringement (unlike here), the court in *Sega* stated: "Our conclusion does not, of course, insulate [defendant] from a claim of copyright infringement with respect to its finished products." *Sega*, 977 F.2d at 1528; *see also Sony*, 203 F.3d at 604 n.7 (final product not alleged to contain infringing material).

Unlike *Sega* and *Sony*, Oracle's API specifications are posted online for anyone to see (though not to copy without a license), so there was no need to copy to discover their functional elements. Moreover, neither *Sega* nor *Sony* concerned rich, complex API specifications like the ones at issue here. These cases dealt with reverse engineering low-level functional requirements.

Sega and *Sony*'s holdings on fair use are distinguishable for another important reason. Copying in those cases was required to understanding functional ideas that were necessary for Accolade's games to work on a Sega Genesis or for PlayStation games to work on Connectix's emulator. That is not the case here, where Google selectively copied to attract programmers. Google did not *have* to copy Oracle's designs. When asked whether Google could have written its own APIs for these 37 packages, Google's expert Owen Astrachan replied: "It's technically possible, but it's not something that could have been done to realize the design goals that I mentioned earlier." (Astrachan Dep. 253:2-4.) The design goals to which Dr. Astrachan referred were leveraging "a large body of software developers" and "a large existing body of code and other libraries." (*Id.* 251:21-24.) This is fundamentally different from the functional compatibility that was required in *Sega* and *Sony* for the complementary products to perform.

Beyond the software arena, this Circuit has enforced copyrights even when the defendant faced strong pressures to adopt a copyrighted work as its own. In *Practice Mgmt. Info. Corp. v. Am. Med. Ass'n*, the Ninth Circuit upheld copyright protection for a coding system for medical procedures even where it became a federal agency's standard, noting that there was nothing to stop the plaintiff's competitors "from developing comparative or better coding systems and lobbying the federal government and private actors to adopt them." 121 F.3d 516, 520 (9th Cir.1997); *see also Del Madera Props. v. Rhodes & Gardner, Inc.*, 637 F. Supp. 262, 264 (N.D. Cal. 1985) (rejecting argument from property developers that governmental approval of copyrighted development plan justified use of map).

Limiting copyright protection for Oracle's API specifications because they have become popular with programmers would punish Oracle for designing effectively. The Ninth Circuit reached a similar conclusion in examining this issue in the trademark context. *See, e.g., Au-Tomotive Gold, Inc. v. Volkswagen of Am., Inc.*, 457 F.3d 1062, 1073 (9th Cir. 2006) (discussing analogous "aesthetic functionality" doctrine in trademark law: "We have squarely rejected the notion that 'any feature of a product which contributes to the consumer appeal and saleability of the product is, as a matter of law, a functional element of that product.'").

IV. THE MERGER DOCTRINE, *SCENE A FAIRE*, AND METHODS OF OPERATION DO NOT SUPPORT GOOGLE'S PROPOSED ADVANCE PROCEDURE.

Google points to the merger, *scenes a faire*, and methods of operation doctrines as supporting its proposed advance determination, but fails to make any showing that they apply. In its summary judgment order, the Court laid out Google's requirements: "If Google believes, for example, that a particular method declaration is a *scene a faire* or is the only possible way to express a given function, then Google should provide evidence and argument supporting its views as to that method declaration." (SJ Order at 9.) Google never does so here.

"Under the merger doctrine, courts will not protect a copyrighted work from infringement if the idea underlying the copyrighted work can be expressed in only one way, lest there be a monopoly on the underlying idea." (*Id.* at 9 (quoting *Satava v. Lowry*, 323 F.3d 805, 812 n.5 (9th Cir. 2003)).) But Google does not give a single example of how there was only one, or very few, ways of expressing the APIs so that the merger doctrine might apply here. To the contrary, "Google does not dispute that, in at least many instances, Sun may have had other choices it could have made when it designed the APIs." (Google Précis at 6.)

Google nonetheless contends it is not possible to distinguish the expression in the 37 API designs from the ideas of the APIs themselves. (*Id.* at 9.) The claim is absurd. The API design specifications are comprised of thousands of elements. No copyright case has ever held this type of extensively detailed expression is simply an "idea." The uncopyrightable ideas here may be the idea of a set of API packages or the idea of an API package containing a set of classes relating to input and output (*e.g.*, java.io). But copyright protects Oracle's particular selection, arrangement, and structure of the 50 classes, 12 interfaces, and 16 exceptions (and all of their constituent member fields and methods) that make up the java.io package, based on choices intended to convey meaning and content to developers. Google copied Oracle's specific expression, not merely the underlying idea.

Google has similarly failed to make any showing that it merely copied *scenes a faire*. Google has again not provided any examples of API elements that consist of "commonplace expressions" that "are indispensable and naturally associated with the treatment of a given idea." (SJ Order at 8-9.) Instead, Google argues that the *scenes a faire* doctrine covers everything Google copied because Google purportedly did so in the name of compatibility.

Google has *scenes a faire* law backwards. The proper test under *scenes a faire* is whether the copyrightholder, Oracle, was constrained in its choice of design, not Google. *Control Data Sys., Inc. v. Infoware, Inc.*, 903 F. Supp. 1316, 1323 (D. Minn. 1995). Google miscites

Computer Associates, which focused on the plaintiff's copyrighted work. 982 F.2d at 710 (stating that court must "examine the structural content *of an allegedly infringed program* for elements that might have been dictated by external factors") (emphasis added); *see also Mitel*, 124 F.3d at 1375 (lower court's proper focus "should have remained upon the external factors that dictated [plaintiff's] selection of registers, descriptions, and values," rather than on whether external factors "justified [defendant's] copying.").

Google cites no contrary Ninth Circuit authority. The footnote Google cites from *Sega* focuses on the plaintiff's primitive code, stating: "Sega's key appears to be functional. It consists merely of 20 bytes of initialization code plus the letters S-E-G-A." 977 F.2d at 1524 n.7. Nor do the two other cases cited in *Goldstein on Copyright* § 2.3.2.1 n.46 support Google. *Atari Games Corp. v. Nintendo of Am. Inc.*, 975 F.2d 832 (Fed. Cir. 1992) actually holds the opposite. In *Atari,* the Federal Circuit applied Ninth Circuit law to uphold the plaintiff's preliminary injunction based on Atari's infringement of its copyright in the key used to unlock Nintendo's video game console. *Id.* at 836-37, 847. Reasoning that the key contained a "unique sequence" of code, and that "External factors did not dictate the design" of Nintendo's key, the Court found that the code was entitled to copyright protection. *Id.* at 840. Thus, while Atari could reverse engineer the code to learn its unprotected features — just like the defendants in *Sony* and *Sega* — it could not replicate the copyrighted code. *Id.* at 844. *Goldstein* also relies upon a quote from *Apple Computer, Inc. v. Microsoft Corp*., which merely recites different policy considerations in copyright law. *See* 799 F. Supp. 1006, 1025-26 (N.D. Cal. 1992) (referring to the "ambitious enterprise" of "[s]triking [a] balance" between copyright's purpose of excluding free riders "who do not pay the cost of creation" with "overly inclusive copyright protection" that "can produce its own negative effects by inhibiting the adoption of compatible standards").

Google's approach to methods of operation is similar. Rather than give a concrete example of a method of operation, it attempts to re-litigate its failed summary judgment argument, contending the APIs as a whole are a method of operation. (*See* Google Précis at 5-6.)

The parties have already argued this issue at length. Google's earlier, fully briefed argument on methods of operation relied on *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807 (1st Cir. 1995), a decision that has never been adopted by the Ninth Circuit and that has expressly been rejected by others. *See Mitel*, 124 F.3d at 1372. *Lotus*'s holding that a "method of operation" is any "means by which a person operates something," *Lotus,* 49 F.3d at 815, would strip copyright protection from many computer programs. Instead, the Ninth Circuit asks "whether, on the particular facts of each case, the component in question qualifies as the expression of an idea, or an idea itself," *Johnson Controls, Inc. v. Phoenix Control Sys., Inc.*, 886 F.2d 1173, 1175 (9th Cir. 1989). The Java API specifications are expression, not merely ideas.

V. ADOPTING GOOGLE'S APPROACH WILL BE TIME CONSUMING AND WILL REQUIRE DUPLICATION AT TRIAL

Google devotes only two sentences to the question of whether its proposal will simplify issues for trial. It will not. Oracle believes there is no real question that the selection and arrangement of the Java API specifications exceed the minimal copyrightability requirements of *Feist*. If it prevails in this advance proceeding, Oracle will still need to explain the selection and arrangement of the API specifications to the jury to show the extent of Google's copying to prove substantial similarity and copying and to support Oracle's claim for damages. And Google, for its part, has not abandoned its fair use defense. Thus, if Google should lose on its uncopyrightable subject matter argument, the Court will be faced at trial with a reprise of the very same set of issues, albeit under a different legal rubric.

Conversely, even if Google were to prevail on all, or a significant portion of, its merger or *scenes a faire* arguments, the copyright claim would still go forward. Google would still be liable for copyright infringement if Oracle could show "virtual identity." *See Apple*, 35 F.3d at 1439. Google proclaims that it "*had* to use the same selection of API elements, and the same hierarchical arrangement and structure," purportedly to achieve compatibility (Google Précis at 8), and Oracle would be entitled to present this same evidence at trial.

Finally, Google's argument rests on its purported need for compatibility. As discussed in section III above, there are serious factual problems with Google's compatibility defense. The defense should be tested with live testimony and cross-examination, not decided on a cold record. Regardless of the outcome, if this proceeding does go forward, witnesses from both sides will then need to be presented to the Court a second time at trial. Google's proposal will squander, not conserve, the Court's time.

Respectfully submitted,

/s/ Michael A. Jacobs

Michael A. Jacobs